中文信息处理 Chinese Information Processing

第二章 作业

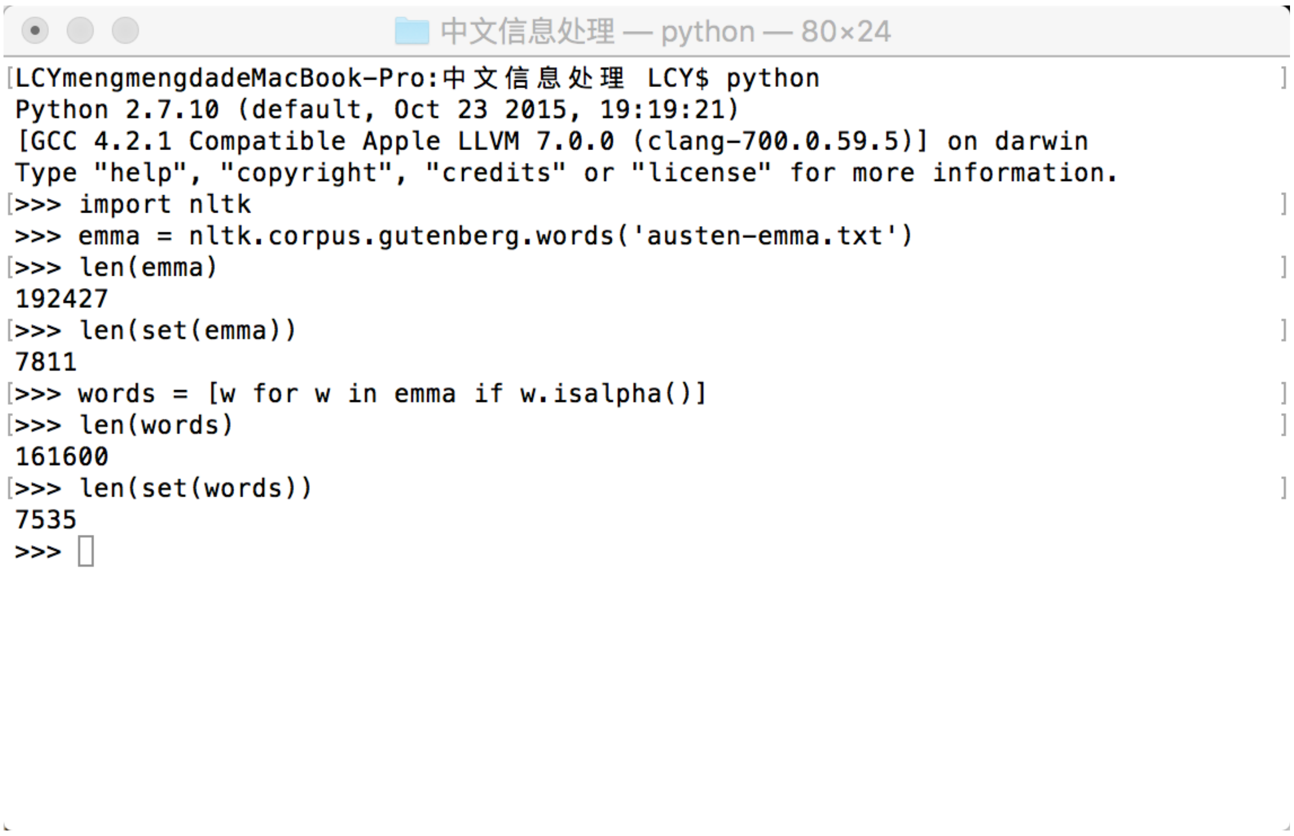
14307130318 刘超颖

1. 使用古腾堡语料库模块处理austen-emma.txt。这本书中有多少个词？多少个不同的词？

```
>>> import nltk
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
>>> len(set(emma))
```

如果将非字母单词去掉，如“,”，“1314”

```
>>> words = [w for w in emma if w.isalpha()]
>>> len(words)
>>> len(set(words))
```

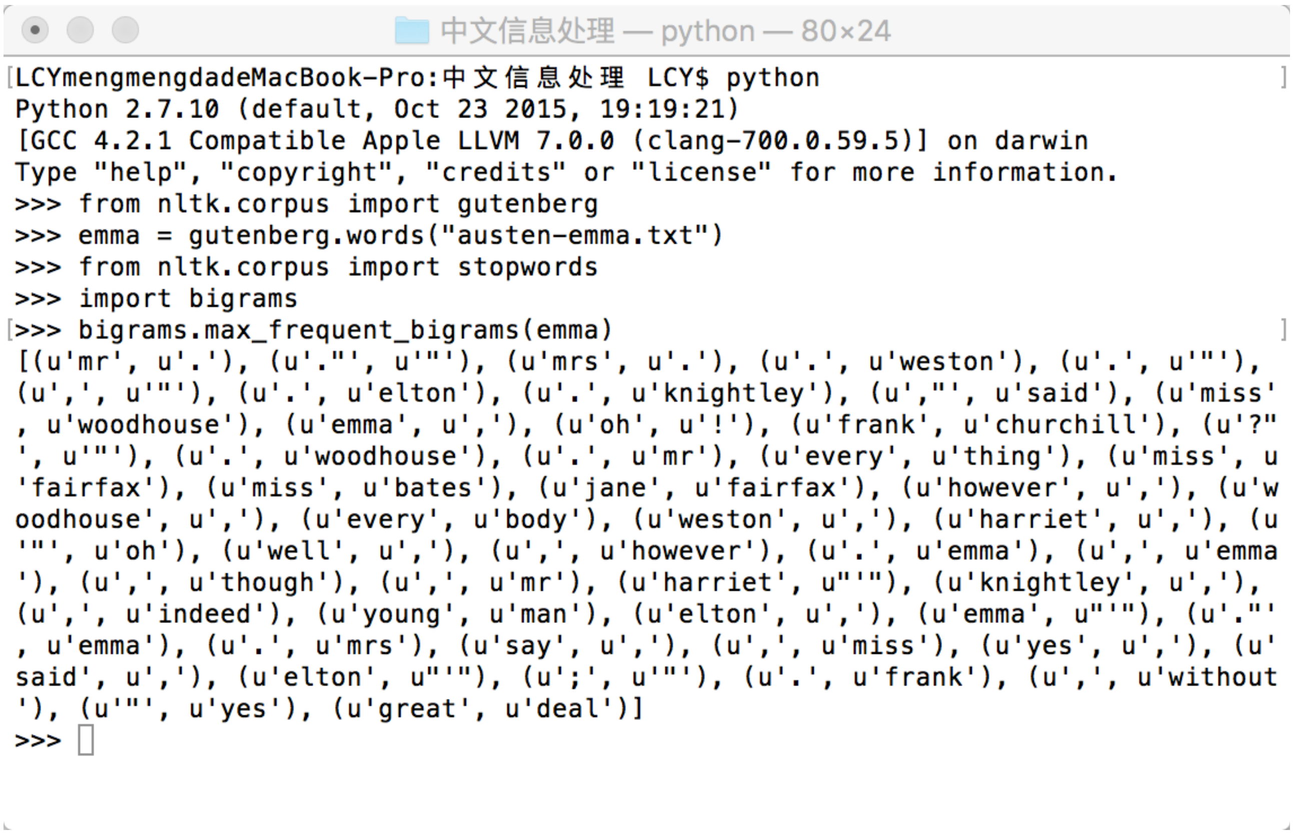


```
LCYmengmengdadeMacBook-Pro:中文信息处理 LCY$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
>>> len(set(emma))
7811
>>> words = [w for w in emma if w.isalpha()]
>>> len(words)
161600
>>> len(set(words))
7535
>>> 
```

2. 写一个程序，输出一个文本中50个最常见的Bigram，忽略包含stopword的bigram。

```
def max_frequent_bigrams(words):
    stopwords_set = set(stopwords.words(fileids = u'english'))
    bigram_words = bigrams([w.lower() for w in words])
    filtered_bigram_words = [bigram_word for bigram_word in bigram
_words if bigram_word[0] not in stopwords_set and bigram_word[1] n
ot in stopwords_set]
    fdist = FreqDist(filtered_bigram_words)
    sorted_filtered_bigram_words = sorted(fdist.keys(), key = lamb
da x:fdist[x], reverse = True)
    return sorted_filtered_bigram_words[:50:]
```

```
>>> from nltk.corpus import gutenberg
>>> emma = gutenberg.words("austen-emma.txt")
>>> from nltk.corpus import stopwords
>>> import bigrams
>>> bigrams.max_frequent_bigrams(emma)
```



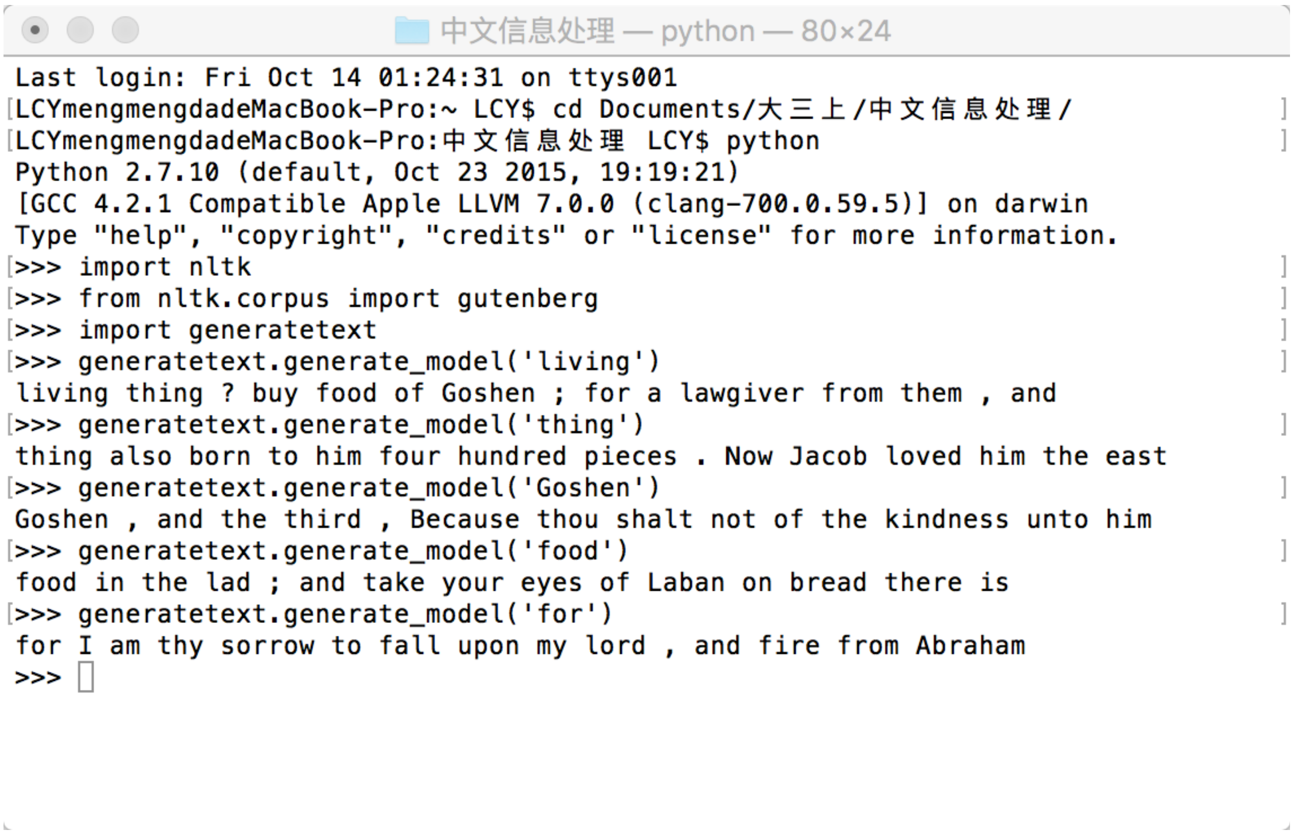
```
LCYmengmengdadeMacBook-Pro:中文信息处理 LCY$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from nltk.corpus import gutenberg
>>> emma = gutenberg.words("austen-emma.txt")
>>> from nltk.corpus import stopwords
>>> import bigrams
>>> bigrams.max_frequent_bigrams(emma)
[(u'mr', u'.'), (u'."', u'"'), (u'mrs', u'.'), (u'.', u'weston'), (u'.', u'"'),
(u',', u'"'), (u'.', u'elton'), (u'.', u'knightley'), (u',"', u'said'), (u'miss'
, u'woodhouse'), (u'emma', u','), (u'oh', u'!'), (u'frank', u'churchill'), (u'?"
', u'"'), (u'.', u'woodhouse'), (u'.', u'mr'), (u'every', u'thing'), (u'miss', u
'fairfax'), (u'miss', u'bates'), (u'jane', u'fairfax'), (u'however', u','), (u'w
oodhouse', u','), (u'every', u'body'), (u'weston', u','), (u'harriet', u','), (u
'"', u'oh'), (u'well', u','), (u',', u'however'), (u'.', u'emma'), (u',', u'emma
'), (u',', u'though'), (u',', u'mr'), (u'harriet', u"'"), (u'knightley', u','),
(u',', u'indeed'), (u'young', u'man'), (u'elton', u','), (u'emma', u"'"), (u'."'
, u'emma'), (u'.', u'mrs'), (u'say', u','), (u',', u'miss'), (u'yes', u','), (u'
said', u','), (u'elton', u"'"), (u';', u'"'), (u'.', u'frank'), (u',', u'without
'), (u'"', u'yes'), (u'great', u'deal')]
>>> 
```

3. 改进随机文本生成程序，选择特定的文体，如：布朗语料库中的一部分或者《创世纪》或者古腾堡语料库中的文本。在此语料上训练一个模型，产生随机文本。可能要实验不同的起始单词。文本的可理解性如何？讨论这种方法产生随机文本的长处和短处。（任选）

```python
import nltk
from nltk import *
import random
from random import choice

def generate_model(word, num = 15):
    text = nltk.corpus.genesis.words('english-kjv.txt')
    for i in range(num):
        bigrams = nltk.bigrams(text)
        print(word),
        link_words = []
        for bigram in bigrams:
            if bigram[0] == word:
                link_words.append(bigram[1])
        word = random.choice(link_words)
```

```
>>> import nltk
>>> from nltk.corpus import gutenberg
>>> import generatetext
>>> generatetext.generate_model('living')
>>> generatetext.generate_model('thing')
>>> generatetext.generate_model('Goshen')
>>> generatetext.generate_model('food')
>>> generatetext.generate_model('for')
```



```
Last login: Fri Oct 14 01:24:31 on ttys001
LCYmengmengdadeMacBook-Pro:~ LCY$ cd Documents/大三上/中文信息处理/
LCYmengmengdadeMacBook-Pro:中文信息处理 LCY$ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> from nltk.corpus import gutenberg
>>> import generatetext
>>> generatetext.generate_model('living')
living thing ? buy food of Goshen ; for a lawgiver from them , and
>>> generatetext.generate_model('thing')
thing also born to him four hundred pieces . Now Jacob loved him the east
>>> generatetext.generate_model('Goshen')
Goshen , and the third , Because thou shalt not of the kindness unto him
>>> generatetext.generate_model('food')
food in the lad ; and take your eyes of Laban on bread there is
>>> generatetext.generate_model('for')
for I am thy sorrow to fall upon my lord , and fire from Abraham
>>> 
```

通过所得结果得知，生成的文本的可理解性不强，只有极少数（以“for”为起始单词）情况下能生成出较为符合英语语法的文本，但生成出的文本内容也不符合逻辑。

这种方法生成随机文本的长处：通过bigram的出现频率使用随机函数体现概率，生成的文本不会出现某几个单词反复循环的情况，且根据概率生成文本比较符合自然规律。

这种方法生成随机文本的短处：效率特别低，需要生成列表缓存bigram，且生成的文本可理解性仍然不强，未考虑自然语言中语法、文本内容、上下文关系等因素。

关于效率问题，由于对python语法仍然不熟悉，需在for循环内每次循环都生成一个bigrams列表。如果改到循环外，第一次循环后bigrams列表会被置空，之后就不再能查找到bigram。但每次循环都生成一个bigrams列表大大影响了程序的效率，造成了很多浪费，这点目前还在查找原因，还不知道如何修改。